Powerful, Professional Databases for Windows CE

by David Shier

EDITOR'S NOTE: *In the first part of this series on databases, David provided an overview of the various database options available for Windows CE. In that article it was indicated that the second part would provide more detail about third-party database programs and the third part would cover the "mini" versions of the powerful enterprise database solutions. Because of the general focus of the Jan/Feb issue, we decided to reverse the order of the second and third installments.*

Two of the largest database vendors, Oracle (www.oracle.com/olite) and Sybase (www.sybase.com/mec), have produced Windows CE versions of their powerful database engines. A database engine is a special type of software that allows programmers to design complex data management applications without having to deal with the details of how to manipulate the data itself. These database engines are not accessible to the end user directly, but are simply building blocks for programmers to use in designing application programs.

Connecting to Host Databases

In virtually all applications for mobile databases, synchronizing data with a central database is a key requirement. However, the level of connectivity needed varies greatly.

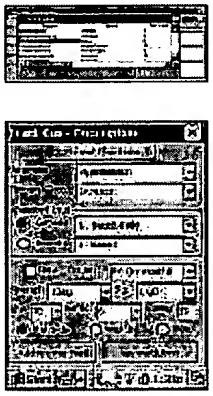
At one end of the spectrum are applications that require nearly constant connection to a host database. An example of this type of application might be an industrial inventory control program. As a user receives parts at the loading dock, the central database is queried to determine the location within the warehouse to store the parts, and the fact that the parts are available is transmitted to the database. For Windows CE handheld PCs a wireless network makes such an application practical. These applications do not need a lot of logic built-into the handheld application because they can rely on the more powerful host computers.

At the opposite end of the connectivity spectrum are applications that must stand on their own except for occasional synchronization. Common applications of this form are field sales force automation programs. The customer list and available products are loaded onto the handheld at the start of the day, and the user works in a "stand-alone" mode entering new orders and updating customer information throughout the day. Then, at the end of the day, the user connects to the central database again and the new information is exchanged.

Of course there are numerous examples of programs that fill the space between these two extremes. My company is currently supporting such a program that fits in the middle of the connectivity range. The application is to allow physicians to write prescriptions electronically. We selected Windows CE because of the wide range of hardware that the operating system supports. Highly mobile doctors can use a Palm-size PC, while those that prefer a keyboard can use a Handheld PC, or even a larger machine.

The program, called ReadyScript, needed to provide considerably more functionality than competing prescription writers. For example, we needed to supply

the doctors with the formularies (lists of medications covered by an insurance plan) for multiple insurance companies (see Screens 1 & 2). We also wanted to supply current clinical guidelines for certain diagnoses. This was an important feature since most physicians are bombarded with information. By providing access to relevant information at the time the doctor needs it, the patients can greatly benefit.



Screens 1 & 2: ReadyScript is a custom prescription-writing application for physicians, developed using a standard database engine.

The problem is, there is far more information in the ReadyScript database than can possibly be stored on a handheld computer. While a constant connection would solve the data issue, it would severely limit the mobility of the ReadyScript tool. This is because we didn't want to connect the handheld computer with a cable, and a constant connection using wireless networks would be far too demanding on the batteries. Instead, we settled on a design that used a database local to the handheld that contained the information most likely to be needed by the user, and a wireless network to retrieve any information that was not expected to be needed. The wireless network is also used to transmit the prescriptions when written. The major advantage of this architecture is that the handheld can operate in a "stand-alone" environment when the wireless network connection is not available.

Regardless of the level of connectivity required, a common feature of handheld databases is "synchronization". In synchronization, the data in the tables residing on the handheld is merged with the data in the host database. This function can be quite complex since some data may have been changed in both locations since the last synchronization session, and there needs to be various rules regarding data accuracy.

For example, in the ReadyScript system, a single physician may enter a prescription for a patient named "John Smith" born on May 28th, 1951, however we have another John Smith in the central database born on the same date. Is this the same patient or just a coincidence? Using a phone number, address, or social security number could resolve the conflict, but what if the patient has moved? Then the name and date of birth would be the same but the address and phone number would have changed. What if the social security number was mistyped? These are issued that are application specific, but representative of the types of problems that occur in distributed database applications.

Such issues are best handled by developing synchronization rules at the host database. In this case, we decided that if everything matched but a single field, then the patient was the same. However, special consideration had to be given to the fact that a father and son may have the same address and phone number, but of course, have different dates of birth, and the son might not have his own social security number. Since these issues are handled at the host database, it is a simple matter to inform the user when a data conflict exists.

Advantage of Relational Databases

If you are familiar with programming, then you might

conclude that the applications we noted could be implemented without the special database engines. This is true, but there are very sound reasons to use Oracle or Sybase. One obvious reason is the fact that maintaining data integrity as noted above, is much easier with the programming tools provided by a database. Another advantage is that that the database engines are designed to allow related data to be found very quickly.

In our ReadyScript example, the doctor needs to be presented with a list of drugs that fit the diagnosis, and are contained on the formulary of the patient's insurance plan. Using a database allows us to assign complex "associations" that make such a query relatively trivial.

Another advantage of using a database engine is that data can be checked for "relational integrity." This means that information entered by the user can be verified to be free from logical errors without complex programming by the developer. For example, if the database contains a table of the insurance plans that are supported by each doctor, then the program can reject entries for patients that don't match a valid insurance company/doctor combination. (In this case, the patient would need to be entered as a "fee-for-service" patient so that everyone is aware of the potential cost to the patient.

Using a Handheld as a Laptop Replacement

The company IMS Health Strategic Technologies (www.imshealth.com/) is a leading supplier of sales force automation tools for the pharmaceutical industry with approximately tens of thousands of users of their laptop computer-based software.

Of course regular readers of *Handheld PC Magazine*

immediately see the value in porting such an application to Windows CE. Not only are handheld PCs much more portable, their extended battery life and instant on features are of great advantage to the mobile salesperson.

IMS Health elected to write the handheld application, called PhasTrak, from scratch, rather than modify their existing Windows 95 application. Of course they leveraged their existing engineering and design in creating the new application. This was made possible since programming for Windows CE uses the same tools and methods as programming for its larger Windows siblings, and by using Sybase's SQL Anyware Mobile database engine.

The IMS Health application is extremely impressive in its complexity. The contact management section (see Screen 3) provides extensive information about physicians, hospitals, medical groups, and even their golf partners!

Screen 3: PhasTrak's Organizational Profile organizes contact information about physicians, hospitals, medical groups and more.

In taking advantage of the Sybase relational database on the handheld, PhasTrak allows the user to rapidly switch between screens of affiliated people and organizations.

According to John Moran, Director of Product Marketing at IMS Health, Sybase provided the technical support needed to assure that current users of their laptop application would find the handheld application familiar. To accomplish this, the engineers at IMS Health needed to use some custom designed

ActiveX controls as well. Looking at the sample screens for the program, you may notice a more complex user interface than is typically available in Windows CE applications.

Capturing Critical Data

One of the most important features of PhasTrak is the logging of sample products provided to physicians. The pharmaceuticals sales representative can provide the physicians with free samples of drugs. However, because these are "controlled substances" there are clear regulations about documenting the disposition of any samples. The doctors must sign an acknowledgment of the transaction and this documentation must be made available for inspection by the FDA upon request. PhasTrak provides the doctor with a screen displaying the samples received and standard regulatory information (see Screen 4).

Screen 4: PhasTrak maintains a log of sample products provided to physicians with a screen displaying the samples received and standard regulatory information.

The doctor signs for the samples on the H/PC screen and this information gets uploaded to the master database for archival purposes. Jay Duff, Vice President of Product Portfolio Management for IMS Health, explained that the need to present all the information on this screen to the physician at the time they sign for the samples is what dictated the use of the H/PC instead of the smaller Palm-Size PC.

What Does All This Mean to You?

Looking at the sample applications highlighted here, it becomes apparent that the old "conventional wisdom"

that Windows CE is not powerful enough just isn't true any longer for almost any mobile application you can imagine. Using Visual C++, ActiveX controls and a database engine from Oracle or Sybase, extremely complex applications can be designed for the current crop of handheld computers.

It's important to remember that, not only are the screens of these machines smaller, requiring different screen layouts than used for laptop or desktop PCs, but Windows CE is a completely different operating system than Windows 95/98/NT. This means that a program written for "standard" Windows may need significant modifications in order to run under Windows CE. Because both screen designs and the underlying code require modifications, it is often easier to simply start over on the handheld design, but using the skills and knowledge gained from the earlier project.

So what we've shown here is that, if you have a database application to be ported to Windows CE, most likely it can be done. The bad news is that it will require software engineering and depending on the complexity, may be considered a major project with cost and time considerations that could be prohibitive if you need custom development.

On the other hand, a number of commercial desktop applications, such as Timeline (project management software) include a database engine imbedded in the product. If the market demand is there, we may start seeing similarly complex applications for Windows CE in the near future.

Programming Issues

Programmers familiar with databases may be interested in knowing some of the limitations of the "light" versions of the database engines available for Windows

CE. Both Sybase and Oracle point out that the database engines are customized at compile time to include only the features that your application uses. This can make the database engine as small as 50KB or as large as 750K (still extremely small.) Obviously, even at their largest size, there has to be features missing. Sybase indicates the following are not supported by their UltraLite version of Adaptive Server Anywhere:

Schema Modification To modify the schema (database layout) you must build a new version of your application.

System table access You can't access the system tables because they don't exist in UltraLite.

Dynamic embedded SQL All SQL (the standard programming language for databases) must be static. That is because the SQL statements get converted to C/C++ code that is compiled as part of your application.

System functions This includes Adaptive Server Anywhere property functions.

Both Oracle and Sybase offer free downloads of their Windows CE database development tools on their respective web sites.

David Shier is the President of Shier Systems & Software, Inc. A former software engineer and Director of Marketing & Sales had specialized for over twenty years on electronic test equipment for the aerospace industry. He is currently authoring a book on automated wire harness testing for aerospace. David started Shier Systems in 1994 to focus on providing ultra-mobile computing and communications solutions including

Windows CE based hardware and software. Shier can be contacted via the company Web site at www.shier.com or by e-mail at shier@shier.com

WindowsCE Webring

Last modified: November 13, 2001